

**CLASS-XII**

**COMPUTER SCIENCE (083)**

**UNIT-III DATABASE MANAGEMENT SYSTEM**

**By: Vikash Kumar Yadav**  
**PGT-Computer Science**  
**K.V. No.-IV ONGC Vadodara**

<b>S. No.</b>	<b>Topic</b>	<b>Page No.</b>
<b>9.1</b>	<b>Introduction</b>	<b>2</b>
<b>9.2</b>	<b>Data Models</b>	<b>2</b>
<b>9.3</b>	<b>Keys</b>	<b>3</b>
<b>9.4</b>	<b>Introduction to MySQL</b>	<b>5</b>
<b>9.5</b>	<b>SQL</b>	<b>6</b>
<b>9.6</b>	<b>SQL Elements</b>	<b>7</b>
<b>9.7</b>	<b>Classification of SQL commands</b>	<b>10</b>
<b>9.8</b>	<b>SQL Queries</b>	<b>10</b>
<b>9.9</b>	<b>Constraints</b>	<b>13</b>
<b>9.10</b>	<b>Adding new column in a table</b>	<b>17</b>
<b>9.11</b>	<b>Modify a table column</b>	<b>17</b>
<b>9.12</b>	<b>Dropping a table</b>	<b>18</b>
<b>9.13</b>	<b>Manipulation data of a table</b>	<b>19</b>
<b>9.14</b>	<b>Making Simple Queries</b>	<b>20</b>
<b>9.15</b>	<b>Queries for special operators</b>	<b>23</b>
<b>9.16</b>	<b>ORDER BY clause</b>	<b>25</b>
<b>9.17</b>	<b>Aggregate functions</b>	<b>25</b>
<b>9.18</b>	<b>Joins</b>	<b>27</b>
<b>9.19</b>	<b>MySQL database connectivity with python</b>	<b>29</b>

## CHAPTER-9 DATABASE MANAGEMENT SYSTEM (DBMS)

### 9.1 INTRODUCTION:

DBMS is a collection of interrelated data in arranged form and set of programs used to access those data.

#### Advantages of DBMS:

- ✓ Control of data redundancy
- ✓ Data consistency
- ✓ Sharing of data
- ✓ Data Integrity
- ✓ Data isolation
- ✓ Privacy and Security

### 9.2 DATA MODELS:

Data models describe the structure of the database. There are four data models in DBMS:

1. Relational Data Model
2. Hierarchical Data Model
3. Network Model
4. Object Oriented Data Model

1. **Relational Data Model:** This database consists of a collection of table. These tables are called relations. A row in a table represents a relationship among a set of values.
  - **Relation:** A relation is a table with columns and rows.
  - **Tuple:** A row of a relation OR a row of a table.
  - **Domain:** A set of values for the columns.
  - **Attribute:** Column name of a relation.
  - **Degree:** Number of attributes in a table.
  - **Cardinality:** Number of tuples OR Number of rows in a table.
  - **View:** A virtual table that does not exists but it is derived from other table.

Roll_No.	St_Name	Fname	Marks	Percentage

2. **Hierarchical Data Model:** In this model the data are represented by collection and relationship between data are represented by links. In this model the collection of data are organized as tree.
3. **Network Data Model:** This is same as the hierarchical model but in this model the collection of data is organized as arbitrary graphs.
4. **Object Oriented Data Model:** In this model the data and its operations are represented by objects.

**NOTE: Some Definitions:-**

**Record:** Collection of attributes

**File:** Collection of records

### 9.3 KEYS:

A key allows us to identify a set of attributes that distinguish entities from each other.

There are four keys:

1. Primary Key
2. Candidate Key
3. Alternate Key
4. Foreign Key

1. **Primary Key:** A primary key is a set of one or more attributes that can uniquely identify tuples within the relation. This key does not have duplicate values in the relation. There must be any value for this key, it cannot be NULL.

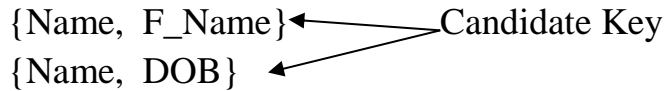
Primary Key



Roll_no.	Name	F_Name	Stream	DOB

**Table: STUDENT**

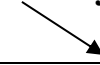
2. **Candidate Key:** All attributes combination inside a relation that can serve as a primary key, is called candidate key.



3. **Alternate Key:** A candidate key that is not the primary key is called an alternate key.

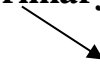
4. **Foreign Key:** A non-key attribute whose values are derived from the primary key of some other table, is known as foreign key in its current table.

Primary Key      **TABLE: CUSTOMER (Parent Table)**



Cust_ID	First_Name	Last_Name

Primary Key      **TABLE: ORDERS (Child Table)**      Foreign Key



Order_No.	Order_Date	Cust_ID	Amount

Primary key of parent table becomes the foreign key for the child table.

## 9.4 INTRODUCTION TO MySQL:

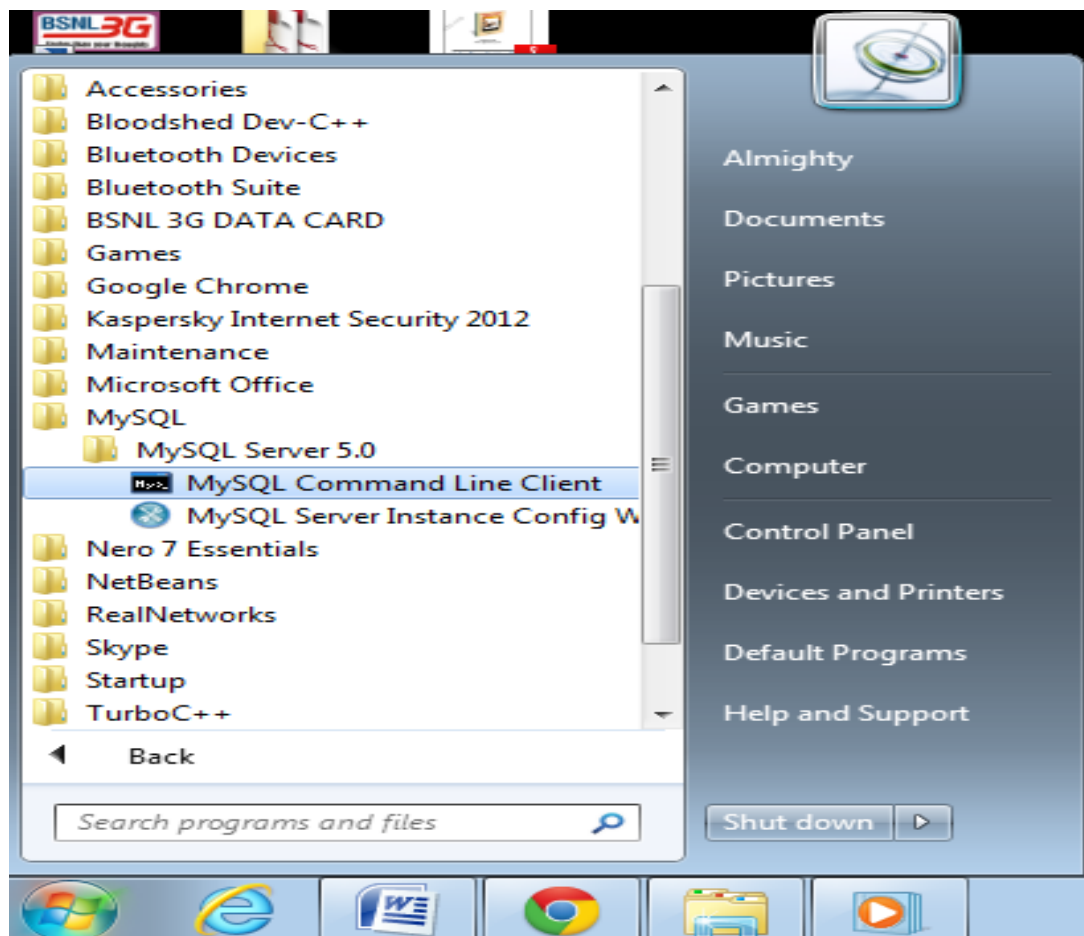
### 9.4.1 Introduction:

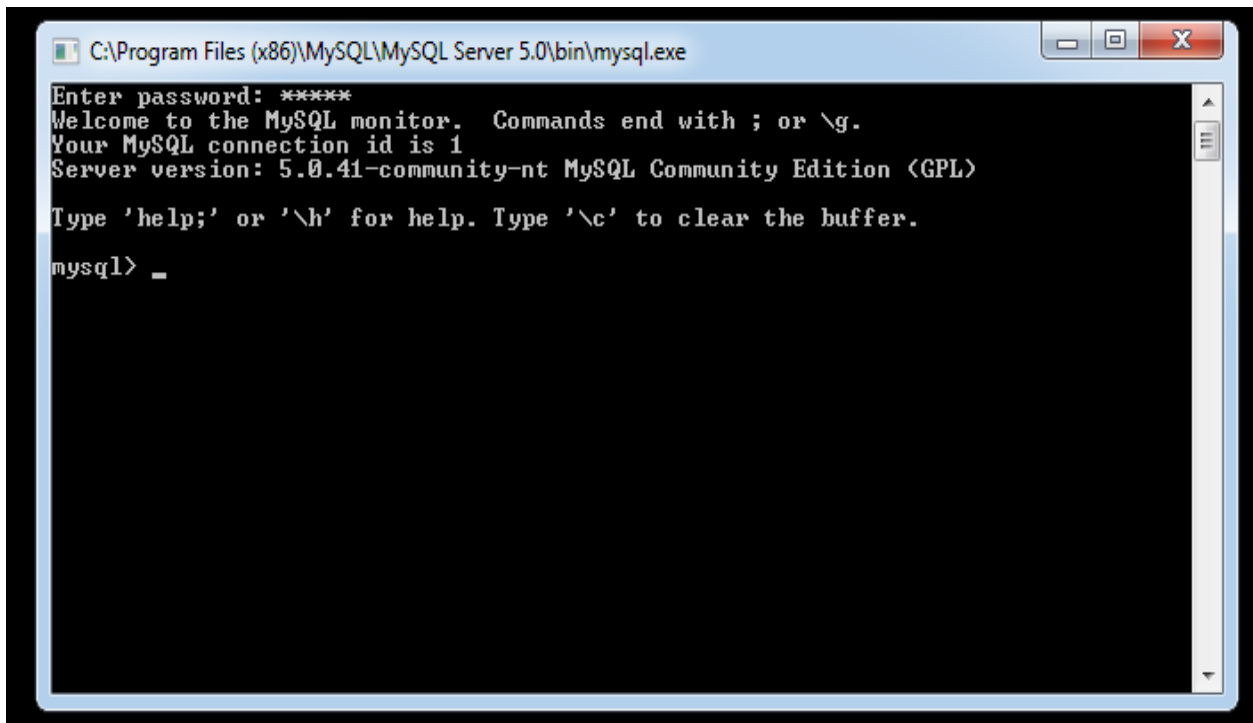
- MySQL stands for My Structured Query Language.
- MySQL is freely available, Open source Relational Database Management System (RDBMS).

### 9.4.2 Features of MySQL:

- Cost: Freely available
- Performance: MySQL is fast.
- Simplicity: MySQL is easy to install.
- High Performance
- Data Security: In has powerful mechanism for ensuring only authorized users have access the data.
- Flexibility

### 9.4.3 STARTING MySQL:





First you have to install MySQL on your system. Then click on

Start → All Programs → MySQL → MySQL Command Line Client

## 9.5 SQL:-

- SQL stands for Structured Query Language.
- This is non-procedural Language.
- This is the common language for relational database. Means this language is used in MySQL.

### Difference between SQL and MySQL:

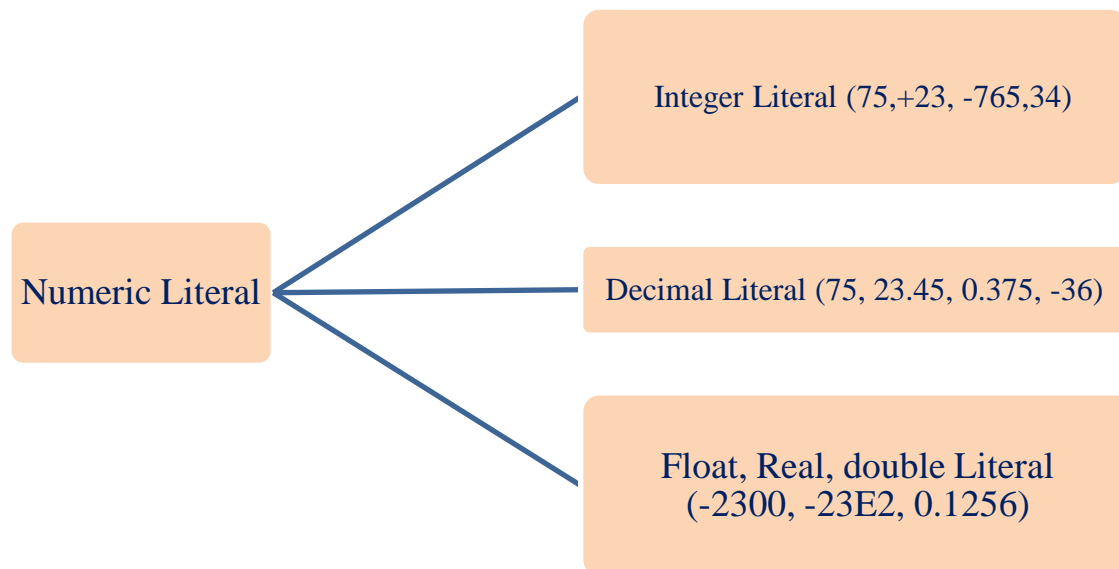
SQL	MySQL
This is the language that used in database.	This is the open source database.
This language is used in MySQL to write the commands in queries.	This is the database. It uses the SQL to write the queries.

## 9.6 SQL Elements:

There are four basic elements of SQL:

1. Literal
2. Data Types
3. Nulls
4. Comments

1. **Literal:** A literal is a fixed data value. This fix value may be numeric or character.



*Character Literal:* All character literals are enclosed in single quotation mark or double quotation marks.

For example :- ‘a’ , “yogesh” , ‘5’ , “xyz5”

2. **Data Types:-** Data types are rules that define what data may be stored in a column and how that data is actually stored. Data types used in MySQL categorized into four categories:

- (i) Numeric
- (ii) String
- (iii) Date and Time
- (iv) Binary

(i) **Numeric:** This data type stores numbers.

Int	Normal size integer that can be signed or unsigned. Width upto 11 digits.
TinyInt	A very small integer that can be signed or unsigned. Width upto 4 digits.
SmallInt	A small integer that can be signed or unsigned. Width upto 5 digits.
MediumInt	A medium size integer that can be signed or unsigned. Width upto 9 digits.
BigInt	A large integer that can be signed or unsigned.
Float(M,D)	A floating point number. M :- Length of total number D :- Number of decimals Ex.- Float(10,2) 10 is the total number of digits and 2 is the number of decimals.
DOUBLE(M,D)	A double precision floating point number.
Decimal(M,D)	Floating point values with varying levels of precision.

(ii) **String:**

Char(M)	A fix length string between 1 to 255 characters. M :- Length
Varchar(M)	A variable length string between 1 to 255 characters.
Enum	Accepts one of a predefined set of strings.
Text	Variable length text with a maximum size of 64 K.
TinyTEXT	Same as text, but with a maximum size of 255 bytes.
MediumTEXT	Same as text, but with a maximum size of 16K.
LongTEXT	Same as text, but with maximum size of 4GB.

(iii) **DATE and TIME:**

DATE	YYYY-MM-DD format. Example: 1998-08-12
DATETIME	YYYY-MM-DD HH:MM:SS format. Example: 1998-08-12 23:37:15
TIMESTAMP	This is same as DATETIME data type, only without hyphens between numbers. YYYYMMDDHHMMSS format. Example: 19980812233715
TIME	Stores the time in HH:MM:SS format.
YEAR(M)	Stores a year in a 2 digits or 4-digits. M :- Length Example: YEAR(2) means 98 YEAR(4) means 1998



- (iv) **Binary Data Type:** Stores the data in binary objects, such as graphics, images, multimedia etc.

BLOB	Stands for Binary Large Objects. Maximum Length=64K
TINYBLOB	Maximum length=255 bytes.
MEDIUMBLOB	Maximum Length= 16MB
LOBLOB	Maximum length= 4GB

3. **NULL values:** If a column in a row has no value, then column is said to be null. NULLs can appear in a column if the column is not restricted by NOT NULL or Primary Key. You should use a null value when the actual value is not known. Null and Zero is not equivalent. Any arithmetic expression containing a null always evaluates to null.

Example:  $7 + \text{null} = \text{null}$

$$7 + 0 = 7$$

} Difference between null and zero.

4. **Comments:** Comment is a text that is not executed. There are two types of comments that we use in MySQL.

(i) Single line Comment: Beginning with two hyphens (--) OR beginning with # symbol.

(ii) Multi Line Comment: `/*.....*/`

**Difference between char and varchar:**

Char	Varchar
1. Fixed length character string.	Variable length character string.
2. When a column is given data type as char(N). If a value is shorter than this length N then blanks are added, but the size of value remains N byte.	Each value that is stored in this column stores exactly as you specify it. No blanks are added, if the length is shorter than maximum length N.
3. Faster to access.	Takes less disk space.

## 9.7 Classification of SQL commands:

SQL commands are categorized into four sub languages:

- (i) Data Definition Language (DDL)
- (ii) Data Manipulation Language (DML)
- (iii) Transaction Control Language (TCL)
- (iv) Data Control Language (DCL)

(i) **Data Definition Language (DDL):** It consist the commands to create objects such as tables, views, indexes etc. in the database.

**COMMANDS:** CREATE, ALTER, DROP, RENAME, TRUNCATE

(ii) **Data Manipulation Language (DML):** It is used for queries. It allows you to perform data manipulation e.g. retrieval, insertion, deletion, modification of data stored in database.

**COMMANDS:** SELECT, INSERT, UPDATE, DELETE

(iii) **Transaction Control Language (TCL):** This language allows you to manage and control the transaction.

**COMMANDS:** COMMIT, ROLLBACK

- NOTE:- Savepoint is also used in TCL.

(iv) **Data Control Language (DCL):** This language is used to control data and access to the databases. It is used for protecting the data from unauthorized access.

**COMMANDS:** GRANT, REVOKE

## 9.8 SQL QUERIES:

In MySQL, to create new tables or query, the user must specify the database. This database is called current database.

**To check the available database in MySQL:-**

SHOW databases;

**To access the database:-**

Syntax:- USE database-name;

Example: USE school;

A database has tables. Tables have rows and columns.

### **To show the available tables in the database:-**

SHOW tables;

**CREATING DATABASE:** In MySQL we can create the databases using CREATE DATABASE statement.

**Syntax:** CREATE DATABASE <database-name>;

**Example:** CREATE DATABASE Bank;

### **OPENING DATABASE:**

**Syntax:** USE <database-name>;

**Example:** USE Bank;

**DROPPING DATABASES:** To remove the entire database we use the DROP DATABASE statement.

**Syntax:** DROP DATABASE <database-name>;

**Example:** DROP DATABASE Bank;

## **Creating Tables in MySQL:**

Tables are created using CTREATE TABLE command.

A table has rows and columns. The column name must be specified along the data type. Each table must have at least one column.

### **Syntax:**

CREATE TABLE <table-name>(<column-name> <data-type> (size), <column-name> <data-type> (size), <column-name> <data-type> (size) );

### **Example:**

```
CREATE TABLE EMPLOYEE(Ecode int(6), Ename varchar(30), Dept varchar(30), city
varchar(25), sex char(1), DOB Date, salary float(12,2) );
```

Ecode	Ename	Dept	City	Sex	Dob	Salary

*Fig. : EMPLOYEE*

**Viewing a table structure:**

```
DESC <table-name>;
```

**Example:** DESC EMPLOYEE;

**Or we can write**

```
DESCRIBE EMPLOYEE;
```

**Inserting data into table:**

The rows(tuples) are added to relations(tables) using INSERT command.

**Syntax:**

```
INSERT INTO <table-name>[<column-name>] VALUES (<value1>, <value2>,
.....);
```

**Example:** INSERT INTO EMPLOYEE VALUES(1001, 'Amit', 'production', 'Ahemedabad', 'M', '1988-08-22', 20000.00);

The **into** clause specifies the target table and the value clause specifies the data to be added to the new row of the table.

While inserting data into tables, following points should be taken care of:

- ✓ Character data should be enclosed within single quotes.
- ✓ NULL values are given as NULL, without any quotes.

- ✓ If no data is available for all the columns then the column list must be included, following the table name. Example: INSERT INTO EMPLOYEE(Ecode, Ename, salary) VALUES(1001, 'Amit', 20000.00);

After inserting the data , we created the following table:

Ecode	Ename	Dept	City	Sex	DOB	Salary
1001	Amit	Production	Ahemedabad	M	1988-08-22	38000.00
1002	Sanjeev	Marketing	New Delhi	M	1990-09-05	32000.00
1003	Imran	RND	Surat	M	1989-01-01	40000.00
1004	Harish	RND	Jaipur	M	1988-01-20	40050.00
1005	Neha	Marketing	Guwahati	F	1985-04-15	35000.00
1006	Dheeraj	Production	Mumbai	M	1984-03-02	39000.00
1007	Vikram	Marketing	Shimla	M	1990-10-10	31000.00
1008	Ashok	Marketing	Patna	M	1980-09-11	40000.00
1009	Priyanka	RND	Gurgaon	F	1990-07-23	40000.00
1010	Seema	Production	New Delhi	F	1989-05-16	37000.00
1011	Manish	Marketing	Guwahati	M	1980-02-07	39050.00

**Table: EMPLOYEE**

## 9.9 CONSTRAINTS:

A constraint is a condition on a field or on set of fields. There are six types of constraints that we use in MySQL:-

- (i) NOT NULL
- (ii) DEFAULT
- (iii) UNIQUE
- (iv) CHECK
- (v) PRIMARY KEY
- (vi) FOREIGN KEY

- (i) **NOT NULL Constraint:** This constraint ensures that a column cannot have NULL value. By default, a column can hold NULL. If you add a NOT NULL constraint on a column it cannot hold a NULL.

## How to apply NOT NULL constraint:-

### (a) When we are creating a table:-

Example:-

```
CREATE TABLE EMPLOYEE (Ecode int(6) NOT NULL, Ename  
varchar(30) NOT NULL, Dept varchar(25));
```

Now Ecode and Ename columns cannot include NULL.

### (b) Add NOT NULL constraint in a existing table column:

Example:-

```
ALTER TABLE EMPLOYEE  
MODIFY Ecode int(6) NOT NULL;
```

(ii) **DEFAULT Constraint:-** The DEFAULT constraint provides a default value to a column. When a user does not enter the value for a column, automatically the defined default value is inserted into the field.

## How to apply default constraint:-

### (a) While we are creating a table:-

Example:-

```
CREATE TABLE EMPLOYEE (Ecode int(6), Ename varchar(30), Salary  
float(12,2)  
DEFAULT 25000.00);
```

### (b) Add DEFAULT constraint in an existing table column:-

Example:-

```
ALTER TABLE EMPLOYEE  
MODIFY Salary float(12,2) DEFAULT 25000.00;
```

(iii) **UNIQUE Constraint:-** It ensures that all values in a column are different.

## How to apply unique constraints:-

### (a) While we are creating a table:-

Example:-

```
CREATE TABLE EMPLOYEE (Ecode int(6) UNIQUE, Ename  
varchar(30), Dept varchar(25));
```

**(b) Add unique constraint in an existing table column:**

Example:-

```
ALTER TABLE EMPLOYEE  
MODIFY Ecode int(6) UNIQUE;
```

- (iv) **CHECK constraint:** - It make sure that all the values in a column satisfy certain criteria.

**How to apply check constraint:-**

**(a) While we are creating a table:-**

Example:-

```
CREATE TABLE EMPLOYEE (Ecode int(11) CHECK(Ecode>0), Ename  
varchar(30));
```

**(b) Add check constraint on an existing table column:-**

Example:-

```
ALTER TABLE EMPLOYEE  
ADD CHECK(Ecode>0);
```

- (v) **Primary Key Constraint:** - This is used to uniquely identify a row in a table.

**How to apply primary key constraint:-**

**(a) While creating a table:-**

Example:-

```
CREATE TABLE EMPLOYEE (Ecode int(6) PRIMARY KEY, Ename  
varchar(30), city varchar(30));
```

**(b) Add primary key constraint on an existing table column:-**

Example:-

```
ALTER TABLE EMPLOYEE  
ADD PRIMARY KEY (Ecode);
```

- (vi) **Foreign Key Constraint:-** Whenever two tables are related by a common column, then the related column in the parent table should be either declared a PRIMARY KEY or UNIQUE KEY and the related column in the child table should have FOREIGN KEY constraint.

**Syntax:-**

Foreign key(<column-to-be-designated-as-foreign-key>) references parent-table (<primary-key-of parent table>);

**Primary Key**

**TABLE: CUSTOMER (Parent Table)**

Cust_ID	First_Name	Last_Name

**Primary Key**

**TABLE: ORDERS (Child Table)**

**Foreign Key**

Order_No.	Order_Date	Cust_ID	Amount

Primary key of parent table becomes the foreign key for the child table.

**(a) While creating a table:-**

Example:-

```
CREATE TABLE ORDERS (Order_no int PRIMARY KEY, Order_Date Date,
Cust_ID integer, Amount double, foreign key(cust_ID) references
CUSTOMER(cust_ID));
```

**(b) Add foreign key constraint through ALTER table:-**

Example:-

```
ALTER TABLE ORDERS
ADD foreign key (cust_ID) references CUSTOMER(cust_ID);
```



## Dropping a constraint:-

The syntax is:-

```
ALTER TABLE table-name  
DROP constraint<constraint-name>;
```

### Example:

To remove primary key constraint:-

```
ALTER TABLE EMPLOYEE  
DROP PRIMARY KEY;
```

To remove foreign key constraint:-

```
ALTER TABLE table1  
DROP FOREIGN KEY fk1;
```

In this query table name is table1 and foreign key constraint is fk1.

## 9.10 ADDING NEW COLUMN IN A TABLE:-

Syntax:-

```
ALTER TABLE table-name  
ADD col-name data-type(size);
```

Example: Add a new column **address** in employee table.

```
ALTER TABLE EMPLOYEE  
ADD address varchar(50);
```

## 9.11 MODIFYING A TABLE COLUMN:

- (i) To change the data-type and size of the column, we use the MODIFY command.
- (ii) To change the name of the column, we use the CHANGE command.

- (i) **To change the data-type and size of the column, we use the MODIFY command:-**

The syntax is:-

```
ALTER TABLE table-name  
MODIFY(Col_name newdatatype(newsize));
```

Example:

```
ALTER TABLE EMPLOYEE  
MODIFY city char(30);
```

- (ii) **To change the name of the column, we use the CHANGE command:-**

```
ALTER TABLE <table-name>  
CHANGE oldcolumnname newcolumnname col-definition;
```

Example:

```
ALTER TABLE EMPLOYEE  
CHANGE Dept Department varchar(30);
```

## 9.12 DROPPING A TABLE:

To remove the entire structure of the table completely, we use the DROP TABLE command.

**Syntax:**

```
DROP TABLE <table-name>;
```

**Example:**

```
DROP TABLE EMPLOYEE;
```

## 9.13 MANIPULATING DATA OF A TABLE:-

- (i) Retrieving data : SELECT command
- (ii) Inserting data : INSERT command
- (iii) Deleting data : DELETE command
- (iv) Modification : UPDATE command

- (i) **SELECT Command:-** A SELECT command retrieves information from the database.
- (ii) **INSERT Command:-** This command is used to insert the data in table.  
NOTE:- We have already discussed about this command.
- (iii) **DELETE Command:-** It means delete the information from the table. This command is used to remove the rows from the table.
  - Specific rows are deleted when you specify the WHERE clause.
  - All rows in the table are deleted if you omit the WHERE clause.

The syntax is:

```
DELETE FROM <table-name>
WHERE <condition> ;
```

Example:- Delete those rows whose department is production.

```
Solution:  DELETE FROM EMPLOYEE
           WHERE dept='production';
```

Example:- Delete all the records of EMPLOYEE table having salary less than 35000.

```
Solution:  DELETE FROM EMPLOYEE
           WHERE Salary<35000;
```

- (iv) **UPDATE command:-** Values of a single column or group of columns can be updated.

The syntax is:-

```
UPDATE table-name
SET column_name=value
WHERE condition;
```

Example:- Change the salary of Vikram to 36000.

```
Solution: UPDATE EMPLOYEE
          SET salary=36000
          WHERE Ename='Vikram';
```

Example:- Increase every employee salary by 10%.

```
Solution: UPDATE EMPLOYEE
          SET salary=salary+(salary*0.1);
```

## 9.14 MAKING SIMPLE QUERIES:-

In SQL queries, we use three clauses mostly:-

- (i) **SELECT:-** What to select
- (ii) **FROM:-** Which Table
- (iii) **WHERE:-** Condition to satisfy

### (i) **SELECT:-**

A SELECT command is used to retrieve information from a table.

- ❖ If you want to select the all columns from a table, then we use the asterisk(\*) in SELECT clause.

Example: -                 SELECT \* FROM EMPLOYEE;

- ❖ To display specific columns of the table by specifying the column names, separated by commas.

Example: -                 SELECT Ecode, Ename, salary  
                              FROM EMPLOYEE;

### (ii) **FROM:-**

A FROM clause, specifies the table name that contains the columns.

### (iii) **WHERE:-**

A WHERE clause, specifies the condition.

```
Syntax:- SELECT column_name
          FROM table_name
          WHERE condition;
```

## SOME IMPORTANT POINTS:-

- ✓ SQL statements are not case sensitive.
- ✓ To end the SQL command, we write the semicolon(;) at the end of a line followed by <ENTER>.

- **Selecting All Columns:-** To select all the columns, we use asterisk (\*) in SELECT statement.

Example:-  
SELECT \*  
FROM EMPLOYEE;

- **Selecting Specific Columns:-** To display the specific columns of the table, write columns name, separated by commas.

Example:-  
SELECT Ecode, Ename, salary  
FROM EMPLOYEE;

- **Eliminating redundant data:-** The DISTINCT keyword eliminates duplicate rows from the results. DISTINCT is used in SELECT statement.

Example:-  
SELECT DISTINCT(Dept)  
FROM EMPLOYEE;

## ALL keyword:-

SQL allows us to use the keyword ALL to specify explicitly that duplicates are not removed.

Example:  
SELECT ALL Dept  
FROM EMPLOYEE;

## Arithmetic Operations:-

The SELECT clause may also contain arithmetic expressions involving the operators +, -, \*, and / operating on constants or attributes.

Example:- Find the new salary of every employee increased by 25%.

```
SELECT Ename,salary,salary*0.25  
FROM EMPLOYEE;
```

**COLUMN ALIAS:-** You can change a column heading by using a column alias.

Example:-  

```
SELECT Ename as Name
FROM EMPLOYEE;
```

### Examples of Queries:-

1. List the name and department of those employees where department is production.

Solution:-  

```
SELECT Ename, Dept
FROM EMPLOYEE
WHERE Dept='production';
```

2. Find the name and salary of those employees whose salary is more than 20000.

Solution:-  

```
SELECT Ename, salary
FROM EMPLOYEE
WHERE salary > 20000;
```

3. Display the name of those employees who live in New Delhi.

Solution:-  

```
SELECT Ename, city
FROM EMPLOYEE
WHERE city='New Delhi';
```

4. List the name of female employees in EMPLOYEE table.

Solution:-  

```
SELECT Ename
FROM EMPLOYEE
WHERE sex='F';
```

5. Display the name and department of those employees who work in surat and salary is greater than 25000.

Solution:-  

```
SELECT Ename, Dept
FROM EMPLOYEE
WHERE city='surat' and salary > 25000;
```

Or we can write this query in another way:

Solution:-  

```
SELECT Ename, Dept
FROM EMPLOYEE
WHERE city='surat' && salary > 25000;
```

6. Display the name of those female employees who work in Mumbai.

Solution:-  

```
SELECT Ename
FROM EMPLOYEE
WHERE sex='F' and city='Mumbai';
```

7. Display the name of those employees whose department is marketing or RND.

```
Solution:- SELECT Ename
           FROM EMPLOYEE
           WHERE Dept='marketing' OR Dept='RND';
```

8. List the name of employees who are not males.

```
Solution:- SELECT Ename, Sex
           FROM EMPLOYEE
           WHERE sex!='M';
```

### 9.15 QUERIES FOR SPECIAL OPERATORS:-

- (i) BETWEEN :- Between two values
- (ii) IN :- Match a value in the list
- (iii) LIKE :- Match a character pattern
- (iv) IS NULL :- Value is null.

#### (i) BETWEEN :-

*Example:-* Find the name and salary of those employees whose salary is between 35000 and 40000.

Solution:-

```
SELECT Ename, salary
FROM EMPLOYEE
WHERE salary BETWEEN 35000 and 40000;
```

Or we can write this query in another way:

```
SELECT Ename, salary
FROM EMPLOYEE
WHERE salary>35000 and salary<40000;
```

#### (ii) IN :-

*Example:-* Find the name of those employees who live in guwahati, surat or jaipur city.

Solution:-

```
SELECT Ename, city
FROM EMPLOYEE
WHERE city IN('Guwahati', 'Surat', 'Jaipur');
```

**(iii) LIKE :-**

% :- It represents any sequence of zero or more characters.

\_ :- Represents any single character.

Example:- Display the name of those employees whose name starts with 'M'.

Solution:-

```
SELECT  Ename
FROM    EMPLOYEE
WHERE   Ename LIKE 'M%';
```

Example:- Display the name of those employees whose department name ends with 'a'.

Solution:-

```
SELECT  Ename
FROM    EMPLOYEE
WHERE   Dept LIKE '%a';
```

Example:- List the name of employees whose name having 'e' as the second character.

Solution:-

```
SELECT  Ename
FROM    EMPLOYEE
WHERE   Ename LIKE '_e%';
```

**(iv) IS NULL :-**

Example:- List the name of employees not assigned to any department.

Solution:-

```
SELECT  Ename
FROM    EMPLOYEE
WHERE   Dept IS NULL;
```

**IFNULL( ) function:-**

If you want to substitute null with a value in the output, you can use IFNULL( ) function.

**Syntax:-**

IFNULL(<column-name>, value to be substitute)



Example:-

```
SELECT   Ename, IFNULL(salary, "5000")
FROM     EMPLOYEE;
```

### 9.16 ORDER BY clause:-

You can sort the result in a specific order using ORDER BY clause. The sorting can be done either in ascending or descending order. The default order is ascending.

Example:- Display the list of employees in descending order of employee code.

Solution:-

```
SELECT   *
FROM     EMPLOYEE
ORDER BY  ecode DESC;
```

Example:- Display the employee code, name in ascending order of salary.

Solution:-

```
SELECT   Ecode, Ename, salary
FROM     EMPLOYEE
ORDER BY  salary asc;
```

Suppose that we wish to list the entire EMPLOYEE relation in descending order of salary. If several employees have the same salary, we order them in ascending order by employee code. We express this query in SQL as follows:-

```
SELECT   *
FROM     EMPLOYEE
ORDER BY  salary desc, Ecode asc;
```

### 9.17 AGGREGATE FUNCTIONS:

Aggregate functions are functions that take a collection of values as input and return a single value. SQL offers five types of aggregate functions:-

- (i) Avg() :- To findout the average
- (ii) Min() :- Minimum value
- (iii) Max() :-Maximum value
- (iv) Sum() :-To calculate the total
- (v) Count() :- For counting

**NOTE:** - The input to sum ( ) and avg( ) must be a collection of numbers, but the other functions can operate on non-numeric data types e.g.string.

**Q.1 Find the average salary of the employees in employee table.**

```
SELECT    avg(salary)
FROM      EMPLOYEE;
```

In some circumstance, we would like to apply the aggregate function not only to a single set of tuples, but also to a group of sets of tuples. We specify this wish in SQL using the group by clause.

The attributes given in the group by clause are used to form groups.

**Example: - Find the average salary at each department.**

Solution: -  
SELECT Dept, avg(salary)  
FROM EMPLOYEE  
group by Dept;

Output for this query

Dept	Avg(salary)
Production	38000.00
Marketing	35410.00
RND	40016.66

**Q.2 Find the minimum salary in EMPLOYEE table.**

Solution:-

```
SELECT    min(salary)
FROM      EMPLOYEE;
```

**Q.3 Find the minimum salary of a female employee in EMPLOYEE table.**

Solution:-

```
SELECT    Ename, min(salary)
FROM      EMPLOYEE
WHERE     sex='F';
```

**Q.4 Find the maximum salary of a male employee in EMPLOYEE table.**

Solution:-

```
SELECT    Ename, max(salary)
FROM      EMPLOYEE
WHERE     sex='M';
```

**Q.5 Find the total salary of those employees who work in Guwahati city.**

Solution:-

```
SELECT      sum(salary)
FROM        EMPLOYEE
WHERE       city='Guwahati';
```

**Q.6 Find the total salary of all employees in EMPLOYEE relation.**

Solution:-

```
SELECT      sum(salary)
FROM        EMPLOYEE;
```

**Q.7 Find the number of tuples in the EMPLOYEE relation.**

Solution:-

```
SELECT      count(*)
FROM        EMPLOYEE;
```

**Q.8 Count the number of employees who work in RND department.**

Solution:-

```
SELECT      count(*)
FROM        EMPLOYEE
WHERE       Dept='RND';
```

## 9.18 JOINS:

An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

Consider two tables, CUSTOMER and ORDERS:

**TABLE: CUSTOMER**

<b>CUST_ID</b>	<b>CUST_NAME</b>	<b>GENDER</b>	<b>CITY</b>
101	Rahul	M	DELHI
105	Sushil	M	GOA
206	Sunita	F	MUMBAI
517	Gaurav	M	VADODARA

**TABLE: ORDERS**

<b>ORDER_ID</b>	<b>CUST_ID</b>	<b>ORDER_DATE</b>	<b>AMOUNT</b>
14578	105	10/04/2016	1850.50
25685	517	15/03/2015	8569.00
89632	222	01/01/2016	5362.20
78451	105	30/05/2015	450.00

## Types of Joins:

1. Cartesian Product
2. Equi Join
3. Natural Join

1. **Cartesian product:** It is known as cross join. The number of tuples in new relation is equal to product of number of tuples of the two tables on which Cartesian product is performed.

```
SELECT Name, Amount
```

```
FROM CUSTOMER, BORROWER;
```

\*\*Cartesian product is formed when no join conditions exist.

2. **Equi Join :** Columns are compared for equality.

There are 4 types of equi join:

- a. INNER JOIN
- b. LEFT JOIN
- c. RIGHT JOIN
- d. FULL JOIN

- a. **INNER JOIN:** Returns all rows when there is at least one match in BOTH tables. This joining is known as simple join. You can use **JOIN** keyword also in the place of **INNER JOIN**. Both will give same result.

### Syntax:

```
SELECT column_name(s)  
FROM table1  
INNER JOIN table2  
ON table1.column_name=table2.column_name;
```

### Example:

```
SELECT CUST_NAME, ORDER_ID  
FROM CUSTOMER  
INNER JOIN ORDERS  
ON  
CUSTOMER.CUST_ID=ORDER.CUST_ID;
```

### OUTPUT

CUST_NAME	ORDER_ID
SHUSHIL	14578
SHUSHIL	78451
GAURAV	25685

- b. **LEFT JOIN:** Return all rows from the left table, and the matched rows from the right table
  - c. **RIGHT JOIN:** Return all rows from the right table, and the matched rows from the left table
  - d. **FULL JOIN:** Return all rows when there is a match in ONE of the tables
3. **Natural Join:** Only one of the identical columns exists.

\*\* The equi join and Natural Join are equivalent except that duplicate columns are eliminated in the Natural Join.

### 9.19 MySQL database connectivity with Python:

- Install python
- Install MySQL
- Install MySQL Driver using following command: (In Command Prompt):

***pip install mysql-connector***

***Note:** Make sure your computer is connected with internet.*

- To verify, whether the connector is properly installed or not, open python shell and type the following command:  
>>>import mysql.connector  
>>>

If the command successfully runs (without any error), then the MySQL connector is successfully installed.

- Now, open MySQL and check the current user, by typing the following command in MySQL:

```
SELECT current_user( );
```

```
Enter password: *****
Welcome to the MySQL monitor.
Your MySQL connection id is 4
Server version: 5.0.41-communit
Type 'help;' or '\h' for help.
mysql> select current_user();
+-----+
| current_user() |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)

mysql>
```

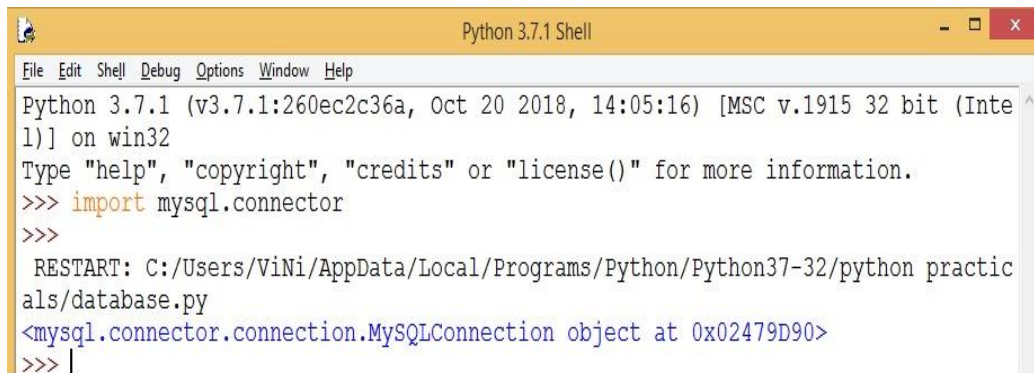
- Connect MySQL database with python. For this, open Python IDLE and write the following code in python file.

**CODE:**

```
import mysql.connector
demodb=mysql.connector.connect(host="localhost",user="root", passwd="computer")
print(demodb)
```

If you get the following output, then the connection made successfully.

**OUTPUT:**



```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import mysql.connector
>>>
RESTART: C:/Users/ViNi/AppData/Local/Programs/Python/Python37-32/python practicals/database.py
<mysql.connector.connection.MySQLConnection object at 0x02479D90>
>>> |
```

- After making successful connection between python and MySQL, now create a database in MySQL through python. For that, write the following code in python:

```
import mysql.connector
demodb = mysql.connector.connect(host="localhost", user="root", passwd="computer")
democursor=demodb.cursor( )
democursor.execute("CREATE DATABASE EDUCATION")
```

- After successful execution of the following code, check in MySQL, whether EDUCATION database has been created or not. for that, write the following command in MySQL:

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| education |
| mysql |
| test |
+-----+
4 rows in set (0.01 sec)

mysql>
```

- If you want to check the created database through python, write the following python code to show the present databases in MySQL.

```
import mysql.connector
demodb = mysql.connector.connect(host="localhost", user="root", passwd="computer")
democursor=demodb.cursor()
democursor.execute("SHOW DATABASES")
for i in democursor:
    print(i)
```

**OUTPUT:**

```
RESTART: C:\Users\ViNi\
als\database.py
('information_schema',)
('education',)
('mysql',)
('test',)
>>> |
```

Here, we can see that EDUCATION database has been created.

**9.19.1 Create a table in database:**

**CODE:**

```
import mysql.connector
demodb = mysql.connector.connect(host="localhost", user="root",
passwd="computer", database="EDUCATION")
democursor=demodb.cursor( )
democursor.execute("CREATE TABLE STUDENT (admn_no int primary key,
sname varchar(30), gender char(1), DOB date, stream varchar(15), marks float(4,2))")
```

To verify the table created or not, write the following code in python:

```
import mysql.connector
demodb = mysql.connector.connect(host="localhost", user="root",
passwd="computer", database="EDUCATION")
democursor = demodb.cursor( )
democursor.execute ("show tables")
for i in democursor:
    print(i)
```

**OUTPUT:**

```
>>>
RESTART: C:\Users\Vinil\
als\database.py
('student',)
>>>
```

### 9.19.2 Insert the data in the table:

```
import mysql.connector
demodb = mysql.connector.connect(host="localhost", user="root",
passwd="computer", database="EDUCATION")
democursor=demodb.cursor( )
democursor.execute("insert into student values (%s, %s, %s, %s, %s, %s)", (1245,
'Arush', 'M', '2003-10-04', 'science', 67.34))
demodb.commit( )
```

### 9.19.3 Fetch the data from table:

```
import mysql.connector
demodb = mysql.connector.connect(host="localhost", user="root",
passwd="computer", database="EDUCATION")
```



```
democursor=demodb.cursor()  
democursor.execute("select * from student")  
for i in democursor:  
    print(i)
```

## OUTPUT:

```
RESTART: C:\Users\ViNi\AppData\Local\Programs\Python\Python37-32\python practic  
als\database.py  
(1245, 'Arush', 'M', datetime.date(2003, 10, 4), 'science', 67.34)  
(1356, 'swati', 'F', datetime.date(2005, 11, 23), 'arts', 72.6)  
>>> |
```

### 9.19.4 Update the record:

```
import mysql.connector  
demodb = mysql.connector.connect(host="localhost", user="root",  
passwd="computer", database="EDUCATION")  
democursor=demodb.cursor()  
democursor.execute("update student set marks=55.68 where admn_no=1356")  
demodb.commit()
```

### 9.19.5 Delete a record:

```
import mysql.connector  
demodb = mysql.connector.connect(host="localhost", user="root",  
passwd="computer", database="EDUCATION")  
democursor=demodb.cursor()  
democursor.execute("delete from student where admn_no=1356")  
demodb.commit()
```